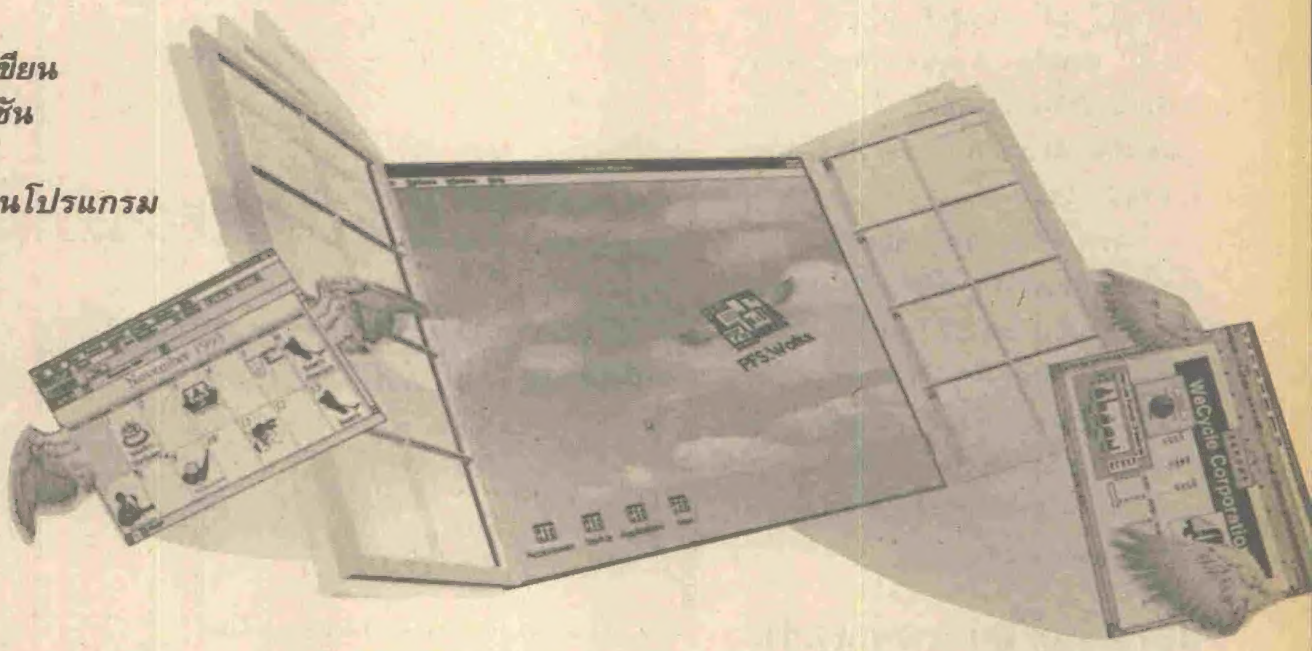
ขจร โรจนเมธินทร์

Visual Basic 3.0
: เอ็มดีไอแอพพลิเคชัน

ศึกษาเทคนิคการเขียน เอ็มดีไอแอพพลิเคชัน ด้วยวิชวลเบสิก เพื่อช่วยให้การเขียนโปรแกรม ทำได้ง่ายขึ้น



กลับมาพบกันอีกครั้งด้วยเรื่องของ Multiple Document Interface (MDI) ซึ่งผู้เขียนเคยแนะนำไปแล้วครั้งหนึ่งในบทความเกี่ยวกับออปเจ็กต์วินโดวส์ของเทอร์โบซีพลัส-พลัสฟอร์วินโดวส์ สำหรับวิชวลเบสิกเองก็มีความสามารถในการสร้างแอพพลิเคชันแบบเอ็มดีไอได้เช่นกัน แถมวิธีการก็ยังแสนจะง่ายดายมากอีกด้วย ผู้อ่านจะเห็นว่าบทความวิชวลเบสิกตอนนี้ผู้เขียนตั้งชื่อว่าเป็นวิชวลเบสิก 3.0 นั้น ความจริงเราสามารถใช้เอ็มดีไอได้ตั้งแต่เวอร์ชัน 2.0 แล้ว แต่ที่เน้นว่าเป็นเวอร์ชัน 3.0 ก็เนื่องจากในบทความตอนต่อ ๆ ไปจะมีการใช้โครงสร้างบางอย่างที่มีเฉพาะในเวอร์ชันนี้เท่านั้น

ความจริงผู้อ่านก็คงคุ้นเคยกับเอ็มดีไอมาแล้วจากแอพพลิเคชันดัง ๆ ต่าง ๆ เช่น ไมโครซอฟต์เวิร์ด ไมโครซอฟต์เอ็กเซล โลตัส 1-2-3 เป็นต้น การใช้เอ็มดีไอนั้นจะทำให้สามารถเปิดด็อกคิวเมนต์ได้หลาย ๆ ชุดพร้อมกัน ดังนั้นการส่งผ่านข้อมูลกันระหว่างด็อกคิวเมนต์จึงทำได้ง่าย สำหรับยูทิลิตี้ของวินโดวส์เองที่ใช้เอ็มดีไอก็มีคือ SYSEDIT.EXE ซึ่งเป็นเอดิเตอร์ที่ใช้แก้ไขคอนฟิกูเรชันไฟล์ หรือตัวโปรแกรมแมนเนเจอร์เองก็จัดเป็นเอ็มดีไอแอพพลิเคชันเช่นกัน ในบทความตอนนี้ผู้เขียนก็จะใช้ SYSEDIT.EXE เป็นตัวอย่างในการเขียนโปรแกรมโดยจะแสดงเป็นขั้นตอนเช่นเคย

ก่อนจะเริ่มสร้าง

เอ็มดีไอแอพพลิเคชันที่สร้างด้วยวิชวลเบสิกนี้จะประกอบด้วยฟอร์มหลัก (Parent form) เพียงฟอร์มเดียว แต่จะมีฟอร์มลูกอยู่ในฟอร์มหลักได้หลาย ๆ ฟอร์ม โดยฟอร์มหลักนี้จะเป็นแบคกราวนด์วินโดวส์ของฟอร์มลูกเสมอ เราจึงไม่สามารถย้ายฟอร์มลูกออกจากฟอร์มหลักได้

ฟอร์มที่จะใช้เป็นฟอร์มหลักนั้นจะต้องเป็นฟอร์มชนิด MDI Form เสมอ ซึ่งจะมีคุณสมบัติต่างไปจากฟอร์มที่เราเคยใช้กัน ส่วนฟอร์มที่จะใช้เป็นฟอร์มลูกจะเป็นฟอร์มธรรมดาที่เคยใช้ สำหรับคอนโทรลอื่นที่จะสามารถสร้างไว้ใน

ฟอร์มหลักได้นั้นจะต้องเป็นคอนโทรลชนิดที่มีคุณสมบัติ Align เท่านั้น ซึ่งในวิชวลเบสิกสแตนดาร์ดเอดิชันจะมีอยู่เพียงอันเดียวคือ Picture box แต่ในโปรเฟสชันแนลเอดิชันจะมีหลายคอนโทรลที่สร้างไว้ในเอ็มดีไอฟอร์มได้ เช่น 3D-Panel เป็นต้น ผู้อ่านอาจคิดว่าหากจำเป็นต้องใช้คอนโทรลชนิดอื่น ๆ ที่ไม่มีคุณสมบัติ Align ในเอ็มดีไอฟอร์มแล้วจะทำอย่างไร

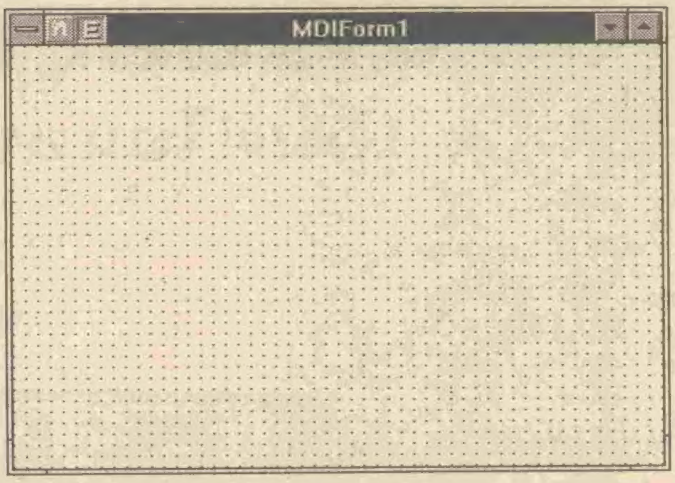
ความจริงเรามีทางเลี่ยงก็คือ ใช้คอนโทรลพิกเจอร์บ็อกซ์นี้เป็นที่เก็บคอนโทรลซึ่งไม่มีคุณสมบัติ Align ที่เราต้องการใช้อีกทีหนึ่ง สาเหตุที่เราไม่สามารถใช้คอนโทรลที่ไม่มีคุณสมบัติ Align ในเอ็มดีไอฟอร์มได้นั้นก็เนื่องมาจากเอ็มดีไอฟอร์มจะต้องคอยจัดการบริเวณไคล์เอ็นต์เอง

สร้างเอ็มดีไอแอพพลิเคชัน

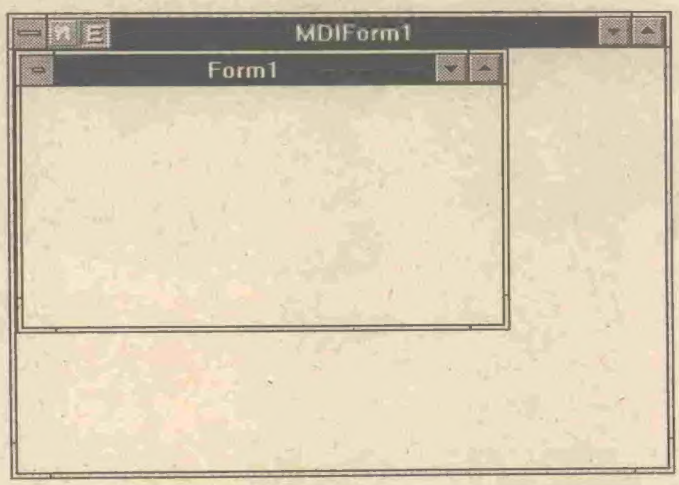
เริ่มต้นผู้อ่านจะต้องสร้างโปรเจ็กต์ใหม่ขึ้นมาก่อน โดยเลือกคำสั่ง New Project จากเมนู File จากนั้นจึงเลือกคำสั่ง New MDI Form ต่อ ก็จะปรากฏเอ็มดีไอฟอร์มตามรูปที่ 1 ซึ่งมี Caption เป็น MDIForm1 และหากลองเลือกเมนู File อีกครั้งก็จะเห็นว่าคำสั่ง New MDI Form ถูกดิสเอเบิลไปแล้ว ดังนั้นเราจึงไม่สามารถสร้างเอ็มดีไอฟอร์มในโปรเจ็กต์เดียวกันได้มากกว่าหนึ่งฟอร์ม สำหรับสีแบคกราวนด์ของเอ็มดีไอฟอร์มนี้จะเห็นว่าเป็นสีเดียวกับวินโดวส์เวอร์กสเปซที่เรากำหนดจาก Control Panel ของวินโดวส์เท่านั้น เนื่องจากไม่มี Property ที่ใช้กำหนดสีแบคกราวนด์ในเอ็มดีไอฟอร์ม

หลังจากสร้างเอ็มดีไอฟอร์มแล้วต่อไปก็จะต้องสร้างชีลด์ฟอร์ม ปกติฟอร์มที่จะใช้เป็นชีลด์ฟอร์มนั้นก็เป็นฟอร์มธรรมดาที่เราเคยใช้กันแล้ว แต่ในที่นี้เราจะต้องบังคับให้ฟอร์มนั้นเป็นชีลด์ฟอร์มโดยการกำหนด Property ค่า MDIChild ของ Form1 ให้มีค่าเป็น True ในขณะที่เราออกแบบนี้ชีลด์ฟอร์มจะยังไม่ถูกย้ายไปอยู่ในเอ็มดีไอฟอร์ม แต่ถ้าผู้อ่านทดลองรันโปรแกรมโดยกด F5 ก็จะพบว่า Form1 จะปรากฏอยู่ใน MDIForm1 เท่านั้น (ดูรูปที่ 2) หากผู้อ่านพยายามเคลื่อนย้าย Form1 ก็จะพบว่าไม่สามารถย้ายออกไปนอกขอบของ MDIForm1 ได้ หรือถ้าลองย่อโปรแกรมให้เป็นไอคอนก็จะพบว่าไอคอนของเอ็มดีไอฟอร์มจะต่างจากฟอร์มทั่วไป ในกรณีที่ขยายชีลด์ฟอร์มพื้นที่ในบริเวณไคล์เอ็นต์จะถูกชีลด์ฟอร์มครอบครองไว้ทั้งหมด และที่ไตเติ้ลบาร์ของหน้าต่างก็จะถูกเปลี่ยนเป็น MDI-Form1 - [Form1]

เนื่องจากชีลด์ฟอร์มถูกจำกัดให้อยู่เฉพาะในเอ็มดีไอฟอร์มเท่านั้น ดังนั้นเมื่อผู้อ่านพยายามเคลื่อนให้ชีลด์ฟอร์มออกไปนอกขอบของเอ็มดีไอฟอร์มก็จะปรากฏสโครลบาร์ขึ้นโดยอัตโนมัติเพื่อที่จะให้เราเลื่อนไปดูบริเวณไคล์เอ็นต์ที่อยู่เกินออกไปจากขอบได้ สโครลบาร์นี้จะมีทั้งในแนวนอนและแนวดิ่งโดยวินโดวส์



รูปที่ 1
แสดงเอ็มดีไอฟอร์ม ขณะเมื่อเริ่มต้นสร้าง



รูปที่ 2
แสดงขณะรัน MDIFor จะมีการโหลด Form1 มาให้ด้วย

จะจัดการให้เอง แต่ถ้าผู้อ่านไม่ต้องการให้มีสโครลบาร์ก็อาจทำได้โดยกำหนดให้ Property ค่า ScrollBars เป็น False สำหรับค่าเริ่มต้นของ MDIForm1 และ Form1 นั้นผู้อ่านจะต้องกำหนดตามตารางที่ 1

สำหรับตัวอย่างโปรแกรมในบทความนี้จะเป็นเอดิเตอร์สำหรับคอนฟิกูเลชันไฟล์ ชื่อ VBSysEdit ดังนั้นเราจะต้องสร้างคอนโทรลเท็กซ์บ็อกซ์เอาไว้ในชีลด์ฟอร์ม Form1 อีกทีเพื่อจะใช้บรรจุเท็กซ์ไฟล์ซึ่งได้แก่ไฟล์ AUTOEXEC.BAT, CONFIG.SYS, SYSTEM.INI และ WIN.INI เราจะต้องเลือกให้เท็กซ์บ็อกซ์สามารถแสดงได้หลายบรรทัดโดยกำหนดให้ MultiLine เป็น True และกำหนดให้มีสโครลบาร์ทั้งสองแนว นอกจากนี้ในชีลด์ฟอร์มจะต้องมีคอนโทรลไทม์เมอร์อยู่ด้วยเพื่อใช้ตรวจสอบว่ามีการเลือกเท็กซ์บ็อกซ์บ้างหรือไม่เพื่อจะได้อีนาเบิลคำสั่งในเมนู Edit ได้ถูกต้อง

ออปเจ็กต์	Property	ค่าเริ่มต้น
MDIForm	Name	fParent
	Caption	"VBSysEdit"
Form	Name	fChild
	Caption	""
	MDIChild	True
Text box	Name	editTxt
	BorderStyle	0-'None
	MultiLine	True
	ScrollBars	3-Both
	Text	""
Timer	Interval	1000

ตารางที่ 1
แสดงค่าเริ่มต้นของคอนโทรลที่เป็นส่วนประกอบของโปรแกรม

เมนูของเอ็มดีไอแอพพลิเคชัน

การสร้างเมนูของเอ็มดีไอฟอร์มและชีลด์ฟอร์มนั้นยังทำเหมือนเดิม โดยกด Ctrl+M เพื่อเรียกวินโดวส์ Menu Design มาใช้ออกแบบเมนู ค่าเริ่มต้นของเมนูจะต้องกำหนดตามตารางที่ 2 สำหรับการสร้างเมนูในเอ็มดีไอแอพพลิเคชันมีเทคนิคที่แตกต่างจากโปรแกรมทั่วไปเล็กน้อย

ประการแรกเมนูที่จะใช้ควบคุมชีลด์ฟอร์มต่าง ๆ ซึ่งโดยมากจะเป็นเมนู Window นั้นจะต้องกำหนดให้ Property ค่า WindowList มีค่าเป็น True เสมอ (ดูรูปที่ 3) เพื่อให้รายชื่อซึ่งเป็น Caption ของชีลด์ฟอร์มที่เปิดอยู่ในบริเวณไคล์เอ็นต์นั้นถูกนำมาแสดงไว้ใน

Caption	Name	Property	Index
&File	fileMain		
&Save	fileSave		
-	fileSep0		
E&xit	fileExit		
-	fileSep1		
&About...	about		
&Edit	editMain		
Cu&t	editCut	Enabled=False, ShortCut=Ctrl+X	
&Copy	editCopy	Enabled=False, ShortCut=Ctrl+C	
&Paste	editPaste	ShortCut=Ctrl+V	
-	editSep		
&Select all	editAll		
&Search	searchMain		
&Find	findWord		
-	findSep		
&Next	findNextWord	Enabled=False, ShortCut=F3	
&Windows	winMain	WindowList=True	
&Cascade	winArrange		0
&Tile	winArrange		1
-	winArrange		2
&Arrange Icons	winArrange		3

ตารางที่ 2
แสดงค่าเริ่มต้นของเมนูที่ต้องสร้างให้ชีลด์ฟอร์ม

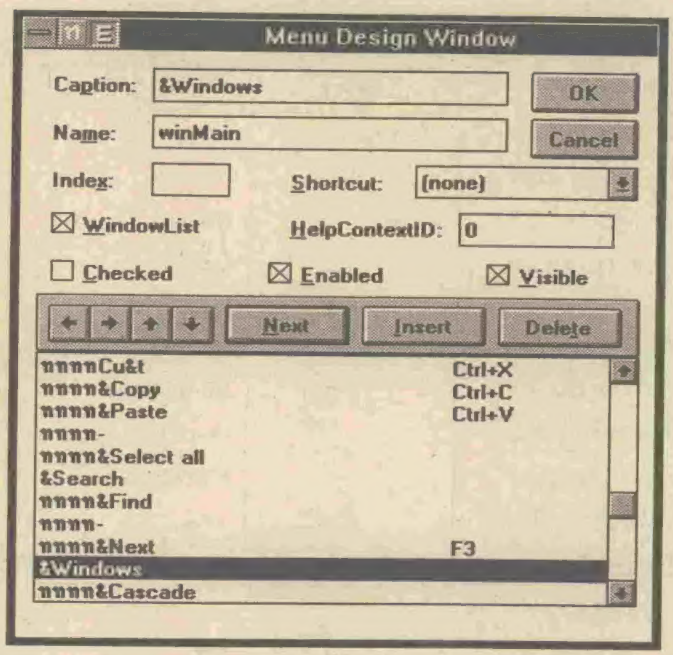
เมนู Window ด้วย ทำให้เราสลับไปมาระหว่างหน้าต่างได้โดยคลิ๊กที่รายชื่อที่ต้องการ

ประการที่สองก็คือ ถ้าผู้อ่านสร้างเมนูให้ฟอร์มหลัก เมนูที่สร้างจะถูกนำมาแสดงบนเมนูบาร์ได้ก็ต่อเมื่อชีลด์ฟอร์มไม่มีเมนู เพราะเมื่อใดที่เราสร้างเมนูให้ชีลด์ฟอร์มแล้ว ขณะที่ชีลด์ฟอร์มถูกโหลด เมนูบาร์ที่แสดงของเอ็มดีไอฟอร์มจะถูกแทนด้วยเมนูของชีลด์ฟอร์มเสมอ ดังนั้นหากต้องการให้คำสั่งใดปรากฏอยู่บนเมนูบาร์ตลอดก็ควรจะต้องออกแบบคำสั่งไว้

จากตารางที่ 2 เป็นค่าเริ่มต้นของเมนูที่เราจะสร้างใน VBSysEdit ในเมนู Edit จะเป็นคำสั่งที่เราจะใช้จัดการกับเท็กซ์ที่เราเลือกไว้เพื่อถ่ายเทผ่านคลิปบอร์ด ดังนั้นค่าเริ่มต้นของคำสั่ง Cut, Copy ต้องกำหนดให้ Property ค่า Enabled เป็น False เพราะขณะเริ่มต้นเท็กซ์จะยังไม่ถูกเลือกแน่นอน นอกจากนี้คำสั่ง Next ในเมนู Find ก็ต้องกำหนดให้ค่า Enabled เป็น False เช่นกัน เพราะขณะเริ่มต้นยังไม่มีการค้นหาคำในเท็กซ์

การจัดการชีลด์ฟอร์มแต่ละอัน

เนื่องจากในเอ็มดีไอแอพพลิเคชันมักจะประกอบด้วยหลาย ๆ ชีลด์ฟอร์ม ดังนั้นเราจึงต้องมีวิธีจัดการกับแต่ละชีลด์ฟอร์มให้ถูกต้อง เราจะถือว่าชีลด์ฟอร์มแต่ละอันเป็นออปเจ็กต์เดี่ยว ๆ ที่ไม่เกี่ยวข้องกันแต่จะเป็นออปเจ็กต์ที่มีลักษณะเหมือนกัน ดังนั้นเราจะจัดการชีลด์ฟอร์มโดยใช้คอนโทรลอะเรย์มาช่วย โดยทั่วไปแล้วในเอ็มดีไอแอพพลิเคชันจะมีจำนวนชีลด์ฟอร์มได้ไม่จำกัด นั่น



รูปที่ 3
แสดงการสร้างเมนู Windo
ของชีลด์ฟอร์มที่กำหนด
WindowList

คือลักษณะคอนโทรลอะเรย์ที่ใช้จะต้องเป็นแบบไดนามิกอะเรย์ แต่ตัวอย่างโปรแกรมในบทความนี้จะจำกัดให้มีชีลด์ฟอร์มได้เพียง 4 ฟอร์มเท่านั้น จึงทำให้ง่ายในการเขียนโปรแกรมเพราะจะใช้คอนโทรลอะเรย์จำนวนคงที่ได้เลย

เราจะประกาศคอนโทรลอะเรย์ของชีลด์ฟอร์มไว้ในโมดูลไฟล์ VBSYSED.BAS ดังนี้

```
Option Explicit
Global frmChild(4) As New fChild
```

จะเห็นว่าเราใช้ชื่อชีลด์ฟอร์ม fChild ที่สร้างขึ้นมากำหนดให้เป็นไทพ์ของออปเจ็กต์ชนิดใหม่โดยใช้คำสั่ง New ช่วย ดังนั้นชีลด์ฟอร์มซึ่งเรากำหนดให้เป็นคอนโทรลอะเรย์แต่ละตัวจะอ้างถึงด้วยชื่อ frmChild(1) ถึง frmChild(4) เพราะฉะนั้นการกระทำใด ๆ กับชีลด์ฟอร์มก็ตามจะต้องระบุอินเด็กซ์ไปด้วยเสมอว่าจะทำกับฟอร์มใด

ตัวแปรและฟังก์ชันเอพีไอที่ต้องประกาศ

ผู้อ่านคงจำได้ว่าผู้เขียนได้เคยเสนอเทคนิคการเขียนโปรแกรมเอดิเตอร์ฉบับย่อด้วยวิชวลเบสิกมาแล้ว โค้ดในโปรแกรม VBSysEdit นี้ส่วนหนึ่งจะดัดแปลงมาใหม่อีกครั้ง สำหรับตัวแปร ค่าคงที่ และฟังก์ชันเอพีไอที่ต้องใช้ ผู้เขียนจะต้องใส่ไว้ในไฟล์ VBSYSED.BAS เช่นกัน

ในรูปที่ 4 จะเห็นว่าผู้เขียนกำหนดตัวแปรที่เป็นอะเรย์ขนาด 4 ตัว ได้แก่ fileList ใช้เก็บชื่อคอนฟิกูเลชันไฟล์ทั้ง 4 ไฟล์, findCase ใช้เก็บแฟล็กที่ใช้ค้นหาสตริงของฟังก์ชัน InStr ว่าจะให้เป็น case-sensitive หรือไม่, findPos ใช้เก็บตำแหน่งของคำที่ค้นหาในเท็กซ์บ็อกซ์ที่พบเป็นครั้งสุดท้ายในแต่ละชีลด์ฟอร์ม และ findTxts จะเป็นคำที่ผู้ใช้ต้องการค้นหาในแต่ละชีลด์ฟอร์ม ผู้อ่านจะเห็นว่าตัวแปรต่าง ๆ ที่ใช้ควบคุมการทำงานของโปรแกรมในชีลด์ฟอร์มแต่ละอันจะต้องแยกจากกันอย่างอิสระ

```
Global fileList(4) As String
Global findCase(4) As Integer
Global findPos(4) As Integer
Global findTxts(4) As String

Global Const MODAL = 1
Global Const MB_YESNOCANCEL = 3
Global Const MB_YESNO = 4
Global Const MB_ICONQUESTION = 32
Global Const MB_ICONEXCLAM = 48
Global Const IDCANCEL = 2
Global Const IDYES = 6
Global Const IDNO = 7
Global Const WM_USER = 1024
Global Const EM_GETMODIFY = WM_USER + 8
Global Const EM_SETMODIFY = WM_USER + 9
Global Const WF_CPU286 = &H2
Global Const WF_CPU386 = &H4
Global Const WF_CPU486 = &H8
Global Const WF_STANDARD = &H10
Global Const WF_ENHANCED = &H20
Global Const WF_80x87 = &H400

Declare Function GetWindowsDirectory% Lib "Kernel" (ByVal lpBuffer As String, ByVal
nSize%)
Declare Function SendMessage& Lib "User" (ByVal hWnd%, ByVal wMsg%, ByVal wParam%,
ByVal lParam&)
Declare Function GetWinFlags& Lib "Kernel" ()
Declare Function GetFreeSpace& Lib "Kernel" (ByVal wFlags%)
```

รูปที่ 4

สำหรับค่าคงที่ต่าง ๆ นั้นจะเป็นค่าคงที่ซึ่งจะใช้กับฟังก์ชัน MsgBox และใช้สำหรับทดสอบส่วนประกอบฮาร์ดแวร์ของเครื่อง ซึ่งเราจะอ่านค่าด้วยฟังก์ชัน GetWinFlags ส่วนฟังก์ชันอื่น ๆ ที่ประกาศไว้ได้แก่ GetFreeSpace จะใช้อ่านค่าหน่วยความจำที่ยังเหลืออยู่, GefWindowsDirectory ใช้ค้นหาชื่อไดเรคทอรีที่ใช้เก็บวินโดวส์, SendMessage ใช้ทดสอบว่าเท็กซ์ที่อยู่ในเท็กซ์บ็อกซ์มีการแก้ไขไปบ้างหรือไม่ เพื่อจะได้เซฟไฟล์ได้ถูกต้อง

โค้ดสำหรับโพรซีเดอร์เหตุการณ์ตอนเริ่มทำงาน

การทำงานของเอ็มดีไอแอพพลิเคชันนั้นอาจเลือกให้เริ่มจากเอ็มดีไอฟอร์มหรือซีลด์ฟอร์มก็ได้ โดยเราสามารถกำหนดได้จากคำสั่ง Project ในเมนู Options โดยกำหนดที่บรรทัด Start Up Form ซึ่งจุดเริ่มต้นการทำงานของโปรแกรมทั้งสองแบบมีความแตกต่างกันเล็กน้อย คือหากผู้อ่านเลือกให้โปรแกรมเริ่มทำงานจากซีลด์ฟอร์ม ขณะที่เริ่มโหลดโปรแกรมทั้งเอ็มดีไอฟอร์มและซีลด์ฟอร์มจะถูกโหลดมาพร้อมกัน แต่ถ้าเลือกให้เริ่มต้นจากเอ็มดีไอฟอร์มโปรแกรมจะไม่โหลดซีลด์ฟอร์มขณะเริ่มทำงานมาให้ สำหรับในโปรแกรม VBSysEdit นี้ผู้เขียนเลือกให้การทำงานเริ่มจาก fParent ซึ่งเป็นเอ็มดีไอฟอร์ม จากนั้นจึงโหลดตัวซีลด์ฟอร์มพร้อมกันทั้งสี่ฟอร์มต่อโดยโพรซีเดอร์เหตุการณ์ MDIForm__Load อีกที

จากข้างต้นที่แสดงรายละเอียดของ MDIForm__Load เริ่มต้นจะเป็นการจัดให้เอ็มดีไอฟอร์มแสดงอยู่กลางจอภาพ โดยกำหนดที่ค่า Left และ Top ของฟอร์ม แล้วจึงเริ่มกำหนดค่าเริ่มต้นอะเรย์ของตัวแปร findCase และ findPos ให้เป็น 1 ต่อจากนั้นก็เรียก InitFileList ซึ่งจะทำหน้าที่กำหนดค่าเริ่มต้นให้ตัวแปร fileList แล้วจึงเริ่มโหลดคอนฟิกูเลชันไฟล์ทั้งสี่ด้วยสับรูทีน LoadFiles

สำหรับโค้ดในรูปที่ 5 ซึ่งเป็นสับรูทีน InitFileList และ LoadFiles นั้นผู้อ่านจะต้องเขียนไว้ในโมดูล VBSYSED.BAS โดย InitFileList จะกำหนดค่าเริ่มต้นให้อะเรย์ fileList ทั้งสี่ตัว สำหรับ fileList(3) และ fileList(4) นั้นจะต้องค้นหาชื่อไดเรคทอรีที่ใช้เก็บวินโดวส์ก่อนแล้วจึงนำมาเชื่อมกับ "SYSTEM.INI" และ "WIN.INI" ตามลำดับ เพื่อจะนำไปใช้โหลดไฟล์ได้ถูกต้อง

สับรูทีน LoadFiles จะทำหน้าที่เปิดไฟล์ทั้งสี่ทีละไฟล์เพื่อจะโหลดลงในเท็กซ์บ็อกซ์ของซีลด์ฟอร์มแต่ละอัน โดยจะต้องเปลี่ยน Caption ของฟอร์มเป็นชื่อไฟล์แต่ละอัน และต้องเก็บค่า

รูปที่ 5

```
Sub InitFileList ()
   Dim I As Integer
   Dim cBuffer As String

   fileList(1) = "C:\AUTOEXEC.BAT"
   fileList(2) = "C:\CONFIG.SYS"
   cBuffer = Space(128)
   I = GetWindowsDirectory(cBuffer, 128)
   cBuffer = Mid(cBuffer, 1, I)
   fileList(3) = cBuffer & "\SYSTEM.INI"
   fileList(4) = cBuffer & "\WIN.INI"
End Sub

Sub LoadFiles ()
   Dim fileHandle As Integer
   Dim I As Integer
   Dim Msg As String

   On Error GoTo loadError

   For I = 1 To 4
    fileHandle = FreeFile
    Open fileList(I) For Input As fileHandle
    frmChild(I).Caption = fileList(I)
    frmChild(I).Tag = I
    frmChild(I).editTxt.Text = Input$(LOF(fileHandle), fileHandle)
    Close fileHandle
   Next
   Exit Sub
loadError:
   Msg = Error$(Err)
   MsgBox Msg, MB_ICONEXCLAM, "VBSysEdit"
   Exit Sub
End Sub
```

รูปที่ 6

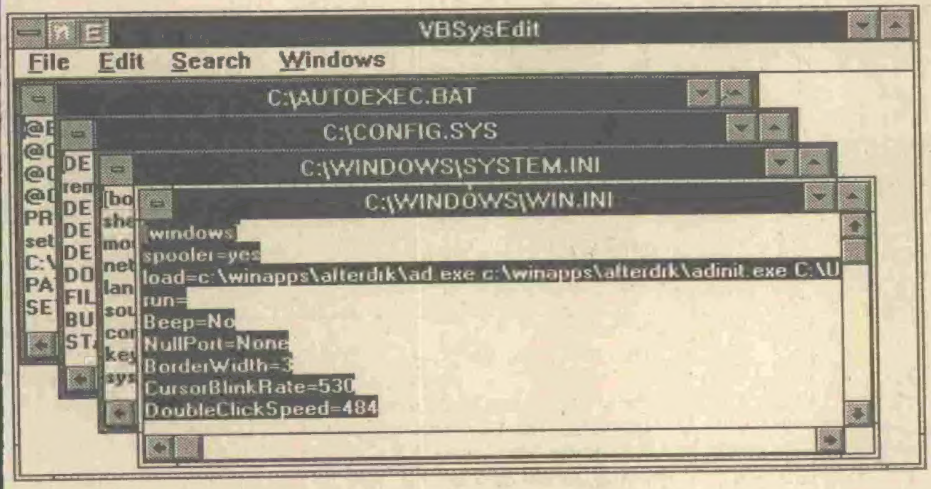
```
Sub editAll_Click ()
   Dim Idx As Integer

   Idx = Val(fParent.ActiveForm.Tag)
   frmChild(Idx).editTxt.SelStart = 0
   frmChild(Idx).editTxt.SelLength = Len(frmChild(Idx).editTxt.Text)
End Sub

Sub editCopy_Click ()
   Dim Idx As Integer

   Idx = Val(fParent.ActiveForm.Tag)
   Clipboard.SetText frmChild(Idx).editTxt.SelText
End Sub

Sub editCut_Click ()
   Dim Idx As Integer

   Idx = Val(fParent.ActiveForm.Tag)
   Clipboard.SetText frmChild(Idx).editTxt.SelText
   frmChild(Idx).editTxt.SelText = ""
End Sub

Sub editPaste_Click ()
   Dim Idx As Integer

   Idx = Val(fParent.ActiveForm.Tag)
   frmChild(Idx).editTxt.SelText = Clipboard.GetText()
End Sub
```

อินเด็กซ์ที่จะใช้อ้างอิงตัวเองของชีลด์ฟอร์มแต่ละฟอร์มไว้ใน Property ค่า Tag เพราะว่าชีลด์ฟอร์มแต่ละตัวจะไม่สามารถรู้จักตัวเองได้เลยถ้าเราไม่เก็บค่านี้ไว้

โค้ดสำหรับเหตุการณ์ในเมนู Edit

เมนู Edit ของโปรแกรม VBSysEdit นี้ไม่ได้กำหนดเมนูเป็นคอนโทรลอะเรย์ ดังนั้นแต่ละเหตุการณ์ที่เกิดขึ้นจึงต้องเขียนแยกกันคนละโพรซีเดอร์ จากโค้ดในรูปที่ 6 จะเห็นว่าเป็นโค้ดที่ใช้จัดการส่งผ่านเท็กซ์ระหว่างคลิปบอร์ดกับโปรแกรมเท่านั้น ซึ่งก็คล้ายกับที่เคยกล่าวไปแล้ว แต่เราต้องไม่ลืมว่าจะต้องจัดการกับเท็กซ์ในชีลด์ฟอร์มแต่ละฟอร์มให้ถูกต้อง ดังนั้นโปรแกรมจะต้องรู้ว่าในขณะนั้นชีลด์ฟอร์มใดที่กำลังแอคตีฟอยู่ ซึ่งก็ทำได้โดยเรียกใช้ Property ค่า ActiveForm ของเอ็มดีไอฟอร์ม ค่าดังกล่าวนี้จะอ่านมาใช้ได้ขณะที่โปรแกรมกำลังทำงานเท่านั้น

เมื่อได้ว่าชีลด์ฟอร์มใดอยู่ในโฟกัสแล้ว เราก็สามารถอ่าน Property ค่า Tag ที่เราเก็บหมายเลขอินเด็กซ์ไว้ดอนโหลดฟอร์มมาใช้ได้ ค่าอินเด็กซ์ที่ได้จะเก็บอยู่ในตัวแปร Idx เมื่อเราได้ค่าอินเด็กซ์ของชีลด์ฟอร์มที่แอคตีฟอยู่แล้ว เราก็จะใช้อินเด็กซ์นี้อ้างไปถึงเท็กซ์บ็อกซ์ของฟอร์มที่ต้องการจากคำสั่ง frmChild(Idx).editTxt สำหรับคำสั่งในเหตุการณ์ editCopy_Click, editCut_Click และ editPaste_Click นั้นจะไม่กล่าวถึงอีก ส่วนเหตุการณ์ editAll_Click ก็เพียงแต่กำหนดให้เท็กซ์ที่เลือกนั้นเป็นเท็กซ์ทั้งหมดที่อยู่ในเท็กซ์บ็อกซ์ โดยกำหนดที่ค่า SelStart



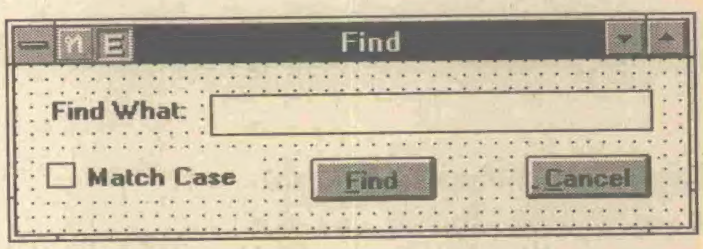
รูปที่ 7 แสดงลักษณะเท็กซ์เมื่อเลือกคำสั่ง Select all ในเมนู Edit

ซึ่งเป็นตำแหน่งเริ่มต้นของเท็กซ์ที่เลือก และ SelLength เป็นความยาวของเท็กซ์ทั้งหมด ดังนั้นเมื่อผู้ใช้ต้องการเลือกเท็กซ์ทั้งหมดก็เพียงแต่เลือกคำสั่งนี้เท่านั้น (ดูรูปที่ 7)

โค้ดที่เกี่ยวข้องกับเมนู Edit อีกส่วนก็คือ โค้ดของคอนโทรลไทม์เมอร์ ซึ่งจะเป็นส่วนที่ใช้ปรับ Property ค่า Enabled ของคำสั่งในเมนู Edit จากโค้ดในรูปที่ 8 เราจะต้องอีนาเบิลคำสั่ง Cut, Copy ให้สามารถใช้งานได้เมื่อมีการเลือกเท็กซ์ โดยเราจะตรวจสอบจากค่า SelText ของซีลด์ฟอร์มที่แอคตีฟอยู่ หากค่า frmChild(Idx).editTxt.SelText ไม่เท่ากับ " " เมื่อไหร่ แสดงว่ามีการเลือกเท็กซ์ในเท็กซ์บ็อกซ์แล้ว คำสั่ง Cut, Copy จึงต้องอีนาเบิล

รูปที่ 8

```
Sub Timer1_Timer ()
   Dim Idx As Integer

   Idx = Val(fParent.ActiveForm.Tag)
   If frmChild(Idx).editTxt.SelText <> "" Then
      frmChild(Idx).editCut.Enabled = True
      frmChild(Idx).editCopy.Enabled = True
   Else
      frmChild(Idx).editCut.Enabled = False
      frmChild(Idx).editCopy.Enabled = False
   End If
End Sub
```



รูปที่ 9 แสดงลักษณะของฟอร์ม fFind

ตารางที่ 3 แสดงค่าเริ่มต้นของฟอร์ม fFind และคอนโทรล

ออปเจ็กต์	Property	ค่าเริ่มต้น
Form	Name	fFind
	BorderStyle	3-Fixed double
	Caption	"Find"
	MaxButton	False
	MinButton	False
Command button	Name	cancelBut
	Caption	"&Cancel"
Command button	Name	findBut
	Caption	"&Find"
	Enabled	False
Check box	Name	findChk
	Caption	"Match Case"
Text box	Name	findTxt
	Text	""
Label	Name	findLbl
	Caption	"Find What:"

โค้ดและฟอร์ม fFind

ฟอร์ม fFind นี้จะใช้รับคำที่ต้องการค้นหาในเท็กซ์ของแต่ละซีลด์ฟอร์ม ตัวฟอร์ม fFind นี้จะไม่เป็นซีลด์ฟอร์ม โครงสร้างของฟอร์มยังเหมือนเดิม (ดูรูปที่ 9) และค่าเริ่มต้นของฟอร์มขอให้สร้างตามตารางที่ 3 เลย

สำหรับโค้ดที่เกี่ยวข้องกับฟอร์ม fFind นั้นตัวฟอร์มจะถูกโหลดจากเมนู Search ด้วยโพรซีเดอร์เหตุการณ์ findWord_Click ส่วนคำสั่ง Next ที่ใช้

```
Sub findWord_Click ()
   fFind.Show MODAL
End Sub

Sub findNextWord_Click ()
   FindWord (findTxts(Val(fParent.ActiveForm.Tag)))
End Sub
```

รูปที่ 10

```
Sub cancelBut_Click ()
   Unload Me
   findPos(Val(fParent.ActiveForm.Tag)) = 1
End Sub

Sub cancelBut_KeyPress (KeyAscii As Integer)
   cancelBut_Click
End Sub
```

รูปที่ 11

```
Sub findBut_Click ()
   Dim Idx As Integer

   Idx = Val(fParent.ActiveForm.Tag)
   FindWord (findTxts(Idx))
   frmChild(Idx).findNextWord.Enabled = True
   Unload Me
End Sub

Sub findBut_KeyPress (KeyAscii As Integer)
   findBut_Click
End Sub
```

รูปที่ 12

```
Sub findChk_Click ()
   Dim Idx As Integer

   Idx = Val(fParent.ActiveForm.Tag)
   If findCase(Idx) = 0 Then
      findCase(Idx) = 1
      findChk.Value = 0
   Else
      findCase(Idx) = 0
      findChk.Value = 1
   End If
End Sub

Sub findTxt_Change ()
   If findTxt.Text <> "" Then
      findTxts(Val(fParent.ActiveForm.Tag)) = findTxt.Text
      findBut.Enabled = True
   End If
End Sub

Sub Form_Load ()
   Left = (Screen.Width - Width) \ 2
   Top = (Screen.Height - Height) \ 2
   findTxt.Text = findTxts(Val(fParent.ActiveForm.Tag))
End Sub
```

รูปที่ 13

ค้นหาคำตำแหน่งถัดไปจะเรียกโพรซีเดอร์เหตุการณ์ findNextWord__Click ซึ่งจะใช้สับรูทีน FindWord ค้นหาคำสำหรับชีลด์ฟอร์มที่แอคตีฟคือ findTxts (Val(fParent.ActiveForm.Tag))) (ดูโค้ดในรูปที่ 10)

จากโค้ดในรูปที่ 11 จะรับผิดชอบต่อบัททอน Cancel โดยปิดฟอร์มด้วยคำสั่ง Unload และคีย์เวิร์ด Me ที่ใช้อ้างถึงตัวฟอร์ม fFind สำหรับโพรซีเดอร์ cancelBut__KeyPress จะรับผิดชอบเมื่อผู้ใช้กดคีย์ Alt + C ก็จะมีผลเหมือนคลิ๊กที่บัททอน

โค้ดสำหรับบัททอน Find จะทำหน้าที่เพียงตรวจดูว่าเหตุการณ์ FindBut__Click ถูกเรียกมาจากชีลด์ฟอร์มใด โดยใช้ Idx เป็นตัวเลือก findTxts (Idx) ซึ่งคือคำที่จะค้นหาสำหรับแต่ละฟอร์มเพื่อส่งให้สับรูทีน FindWord ต่อ

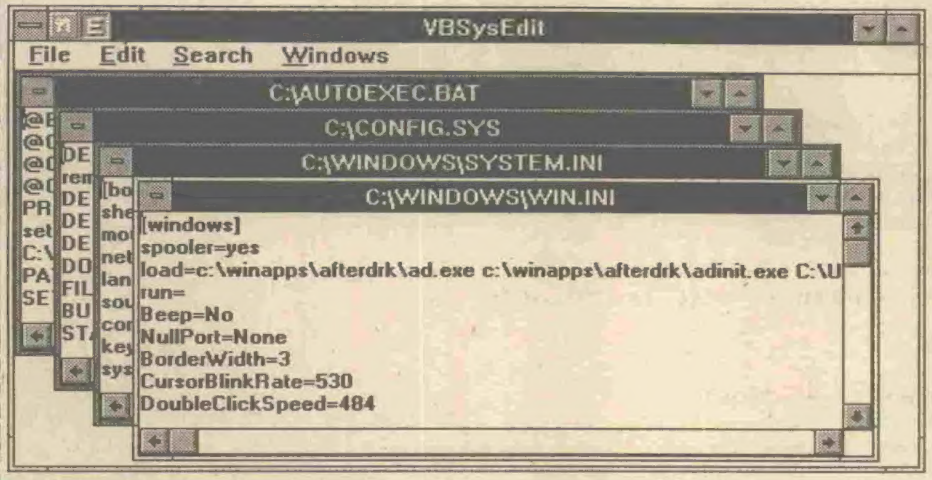
สำหรับโค้ดส่วนที่เหลือซึ่งจะต้องเขียนในฟอร์ม fFind นั้นผู้เขียนแสดงไว้ในรูปที่ 13 ซึ่งประกอบด้วย findChk__Click ซึ่งรับผิดชอบการปรับค่าเช็ค-บ็อกซ์ findChk เมื่อผู้ใช้คลิ๊กเม้าส์, findTxt__Change ใช้เก็บคำที่ผู้ใช้ต้องการค้นหาในแต่ละชีลด์ฟอร์มไว้ในอะเรย์ findTxts ให้ถูกต้อง

การจัดชีลด์ฟอร์มบนบริเวณไคล์เอ็นต์

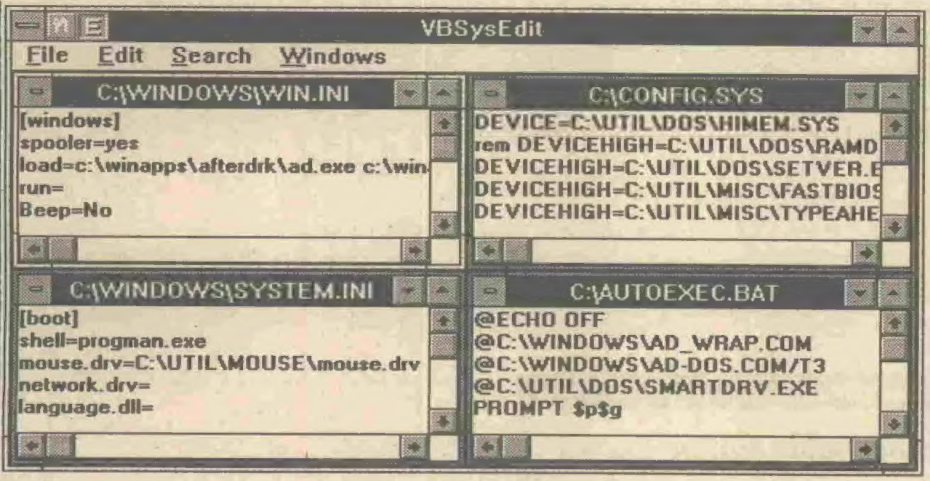
ชีลด์ฟอร์มแต่ละอันจะแสดงอยู่บนบริเวณไคล์เอ็นต์ของฟอร์มหลักได้ 3 แบบ คือ Cascade วิธีนี้ชีลด์ฟอร์มแต่ละอันจะเรียงซ้อนกันตามลำดับจากซ้ายไปขวา (ดูรูปที่ 14), Tile เป็นวิธีที่บริเวณไคล์เอ็นต์ทั้งหมดจะถูกแบ่งกันเพื่อใช้แสดงชีลด์ฟอร์ม (ดูรูปที่ 15) แสดงเป็นไอคอน

การจัดชีลด์ฟอร์มในโปรแกรม VBSysEdit จะทำผ่านเมนู Window ในวิชวลเบสิกนั้นเราสามารถใช้คำสั่ง Arrange จัดชีลด์ฟอร์ม จากโค้ดข้างล่างนี้ เราเพียงแต่ส่งวิธีการที่ต้องการจัดไปด้วย ในที่นี้เราสามารถใช้ค่า Index ของคอนโทรลอะเรย์ส่งไปได้เลย นั่นคือ ถ้าเป็น 0 ก็จะเป็นการจัดแบบ Cascade, 1 เป็นการจัดแบบ Tile และ 3 เป็นการจัดไอคอน

```
Sub winArrange__Click (Index As Integer)
    fParent.Arrange Index
End Sub
```



รูปที่ 14 แสดงการจัดชีลด์ฟอร์มแบบ Cascade บนบริเวณไคล์เอ็นต์



รูปที่ 15 แสดงการจัดชีลด์ฟอร์มแบบ Tile บนบริเวณไคล์เอ็นต์

รูปที่ 16

```
Sub Form_QueryUnload (Cancel As Integer, UnloadMode As Integer)
  Dim Idx As Integer

  Idx = Val(Me.Tag)
  If TextModified(Me.editTxt) Then
    Select Case PromptSave(Idx)
      Case IDYES
        CloseFiles (Idx)
      Case IDCANCEL
        Cancel = True
    End Select
  End If
End Sub
```

ต้องเซฟไฟล์ใดบ้างก่อนออกจากโปรแกรม

เนื่องจากโปรแกรม VBSysEdit ก็จัดเป็นเอดิเตอร์ตัวหนึ่ง ดังนั้นจึงจำเป็นที่โปรแกรมต้องตรวจสอบให้ได้ว่ามีการแก้ไขเท็กซ์ในตัวเอดิเตอร์บ้างหรือไม่ ถ้ามีก็จะต้องถามผู้ใช้ว่าต้องการจะเซฟไฟล์ก่อนออกจากโปรแกรมหรือไม่ ดังนั้นก่อนที่ชีลด์ฟอร์มแต่ละฟอร์มจะปิดตัวเอง เราจะต้องเขียนโปรแกรมเพื่อให้ชีลด์ฟอร์มแต่ละตัวตรวจสอบตัวเองได้ก่อนว่าเท็กซ์ในเท็กซ์บ็อกซ์ถูกแก้ไขไปแล้วและจำเป็นต้องเซฟ

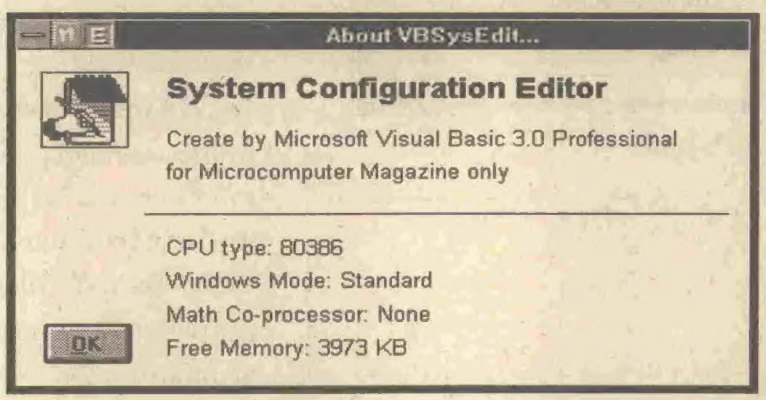
จากโค้ดในรูปที่ 16 จะเห็นว่าเราใช้เหตุการณ์ Form_QueryUnload ซึ่งจะเกิดขึ้นก่อนที่ชีลด์ฟอร์มแต่ละฟอร์มจะถูกปิด ดังนั้นเมื่อเกิดเหตุการณ์นี้ขึ้นเราจึงสามารถตรวจสอบก่อนได้ว่าเท็กซ์ของแต่ละชีลด์ฟอร์มถูกแก้ไขไปหรือยังโดยใช้ฟังก์ชัน TextModified ตรวจดูแฟล็กของเท็กซ์บ็อกซ์ ถ้าหากมีการแก้ไขโปรแกรมก็จะสอบถามว่าต้องการเซฟไฟล์หรือไม่ หากต้องการเซฟก็เรียก CloseFiles แต่ถ้าไม่ต้องการเซฟจะต้องเซตให้ตัวแปร Cancel เป็น True เพื่อจะยกเลิกการปิดชีลด์ฟอร์มนั้น

โค้ดในรูปที่ 17 จะแสดง CloseFiles ที่ทำหน้าที่เซฟเท็กซ์ในเท็กซ์บ็อกซ์ที่ต้องการลงไฟล์ โดยเราจะส่งค่าอินเด็กซ์ของชีลด์ฟอร์มที่ต้องเซฟมาให้ เพื่อที่จะได้ใช้ชื่อไฟล์ที่จะเซฟได้ถูกต้อง สำหรับฟังก์ชัน PromptSave, TextModified และ TextClean นั้นผู้เขียนเคยกล่าวถึงในบทความตอนก่อนแล้ว จึงขออนุญาตไม่กล่าวถึงอีก

รูปที่ 17

```
Sub CloseFiles (Idx As Integer)
   Dim fileHandle As Integer
   Dim response As Integer

   On Error GoTo CloseError
   fileHandle = FreeFile
   Open fileList(Idx) For Output As fileHandle
   Print #fileHandle, frmChild(Idx).editTxt.Text
   Close fileHandle
   Exit Sub
CloseError:
   MsgBox Error$(Err), MB_ICONEXCLAM, "VBSysEdit"
   Exit Sub
End Sub

Function PromptSave% (I As Integer)
   Dim Msg As String
   Dim nl As String

   nl = Chr(13) + Chr(10)
   Msg = "The Text in the " & fileList(I) & " file has changed." + nl + nl
   Msg = Msg & "Do you want to save changes?"
   PromptSave = MsgBox(Msg, MB_ICONQUESTION + MB_YESNOCANCEL, "VBSysEdit")
End Function

Sub TextClean (TBox As Control)
   Dim Ok As Long
   Ok = SendMessage(TBox.hWnd, EM_SETMODIFY, 0, 0&)
End Sub

Function TextModified (TBox As Control)
   TextModified = SendMessage&(TBox.hWnd, EM_GETMODIFY, 0, 0&) * -1
End Function
```

รูปที่ 18
แสดงฟอร์ม fAbout
เมื่อโหลดจากคำสั่ง
About ในเมนู File



เทคนิคการสร้างบ็อกซ์ About แบบใหม่

ในบทความตอนนี้เราจะเพิ่มรายละเอียดให้กับบ็อกซ์ About ให้มีมากขึ้นกว่าเดิม จากรูปที่ 18 จะแสดงลักษณะของฟอร์ม fAbout ซึ่งจะแสดงชนิดของซีพียูที่ใช้ โหมดการทำงานของวินโดวส์ การติดตั้งโคโปรเซสเซอร์ จำนวนหน่วยความจำที่ยังเหลืออยู่เป็นกิโลไบต์ ก่อนอื่นเราจะต้องเริ่มสร้างฟอร์มกันก่อนโดยใช้ค่าเริ่มต้นจากตารางที่ 4

หลังจากสร้างฟอร์มตามตารางที่ 4

ารางที่ 4

อปเจ็กต์	Property	ค่าเริ่มต้น
Form	Name	fAbout
	BorderStyle	3-fixed double
	Caption	"About VBSysEdit"
	MaxButton	False
	MinButton	False
Command Button	Name	aboutOKbut
	Caption	"&OK"
Label	Name	Label1
	Caption	"System Configuration Editor"
Label	Name	Label2
	Caption	"Create by Microsoft Visual Basic 3.0 Professional"
Label	Name	Label7
	Caption	"for Microcomputer Magazine only"
Label	Name	Label3-Label7
	Caption	""
Image	Name	Image1
	Strech	True

```
Sub aboutOKbut_Click ()
   Unload Me
End Sub

Sub aboutOKbut_KeyPress (KeyAscii As Integer)
   Unload Me
End Sub

Sub Form_Load ()
   Dim Flags As Long

   Left = (Screen.Width - Width) \ 2
   Top = (Screen.Height - Height) \ 2
   Flags = GetWinFlags()
   Match = 1
   Select Case Match
     Case (Flags And WF_CPU486) \ WF_CPU486
       label3.Caption = "CPU type: 80486"
     Case (Flags And WF_CPU386) \ WF_CPU386
       label3.Caption = "CPU type: 80386"
     Case Else
       label3.Caption = "CPU type: 80286"
   End Select
   If Flags And WF_ENHANCED Then
     label4.Caption = "Windows Mode: 386 Enhanced"
   Else
     label4.Caption = "Windows Mode: Standard"
   End If
   If Flags And WF_80x87 Then
     label5.Caption = "Math Co-processor: Present"
   Else
     label5.Caption = "Math Co-processor: None"
   End If
   label6.Caption = "Free Memory: " + Format$(GetFreeSpace(0) \ 1024) + " KB"
End Sub
```

รูปที่ 19

แล้ว ก็จะต้องเขียนโค้ดสำหรับฟอร์ม fAbout จากโค้ดในรูปที่ 19 จะมีโค้ดสำหรับเหตุการณ์ aboutOKbut_Click, aboutOKbut_KeyPress ที่รับผิดชอบสำหรับบัททอน OK ที่จะปิดฟอร์ม fAbout ส่วนการทำงานอื่น ๆ นั้นจะทำตอนฟอร์มถูกโหลด เราจะเขียนโค้ดไว้ในเหตุการณ์ Form_Load โดยจัดให้ฟอร์มแสดงอยู่ตรงกลางจอ จากนั้นก็ใช้ฟังก์ชัน GetWinFlags อ่านค่าแฟล็กของวินโดวส์ แล้วนำค่าที่ได้มาทดสอบว่าซีพียูของเครื่องที่ใช้งานอยู่ขณะนั้นเป็นชนิดใด ทดสอบโหมดการทำงานของวินโดวส์ ทดสอบว่ามีการติดตั้งโคโปรเซสเซอร์หรือไม่ หลังจากนั้นจึงอ่านค่าจำนวนหน่วยความจำที่เหลือด้วยฟังก์ชัน GetFreeSpace และคำนวณเป็นกิโลไบต์ ค่าทั้งหมดที่ทดสอบได้จะถูกนำมาแสดงในฟอร์ม fAbout ที่ Caption ของเลเบลแต่ละอันที่กำหนดไว้แล้ว

ก่อนจะจบตอนนี้

ผู้เขียนเห็นว่าการเขียนเอ็มดีไอแอพพลิเคชันด้วยวิชวลเบสิกนั้นมีการจัดการที่ง่าย ทำให้การเขียนโปรแกรมทำได้ง่าย เทคนิคการเขียนโปรแกรมเอ็มดีไอนี้ผู้อ่านสามารถนำไปใช้ในการสร้างแอพพลิเคชันต่อได้ โดยอาจนำไปใช้ในการสร้างเอดิเตอร์ที่สนับสนุนเอ็มดีไอแบบแท้จริงก็ได้ การใช้ไดนามิกคอนโทรลอะเรย์โดยคำสั่ง ReDim, ReDim Preserve และ UBound จะช่วยให้ผู้อ่านสามารถควบคุมชีลด์ฟอร์มได้โดยไม่จำกัดจำนวนอีกต่อไป ทดลองดูสิครับ